自主导航小推车操作指南及教程

中文版（Chinese）

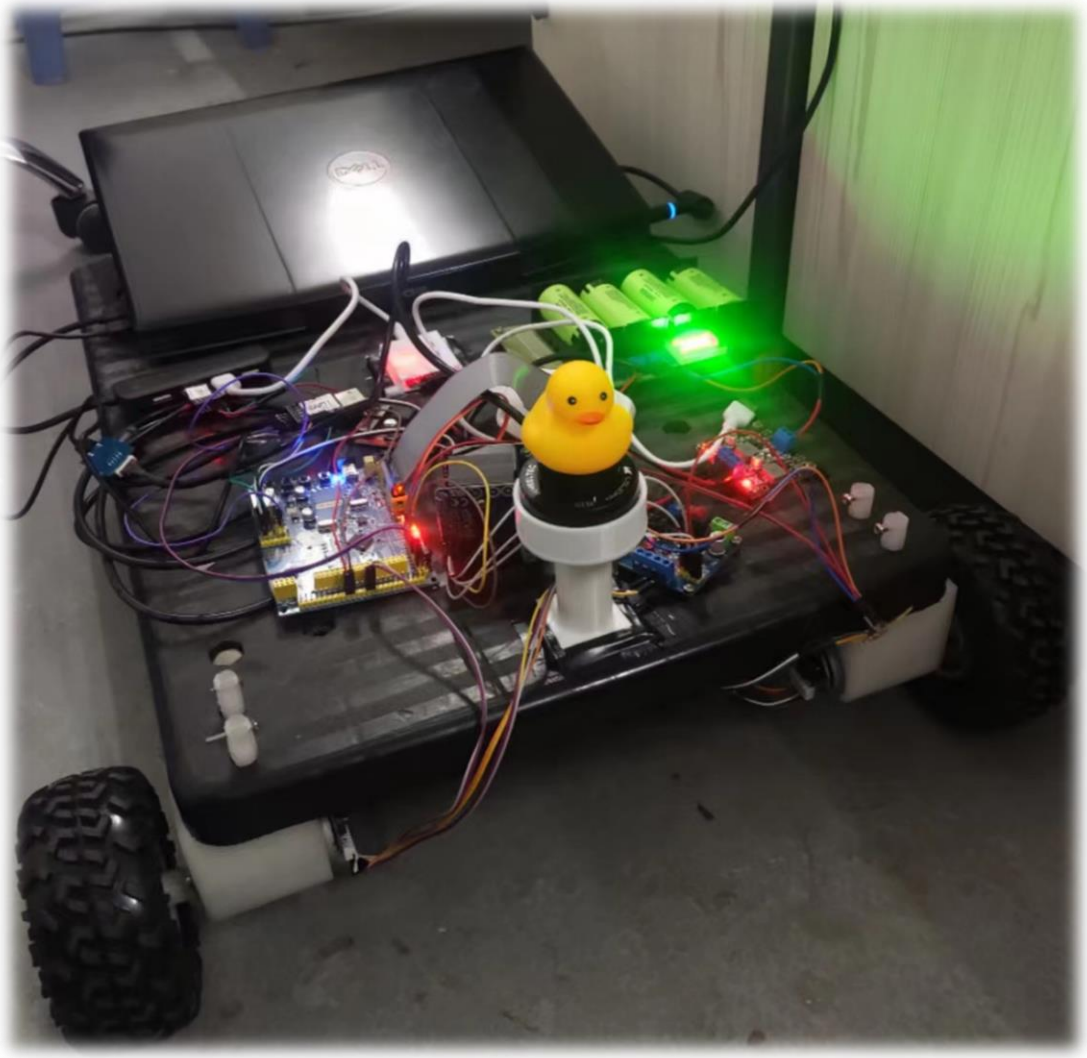
作者：陈景铨、赖伟俊

版本：V1.2 版本

开始时间：　2022.10

最近更新时间：2024.7.14

更新内容：更新了传感器的购买链接；合并了 32 端移植教程；对各部分进行了更详细的介绍。



目录

1.前言介绍

本项目是一辆 ROS 自主导航小推车。该 ROS 机器人将小推车作为底盘，并装载了两个直流电机以产生动力，实现了自主定位，导航，slam，opencv 等功能。

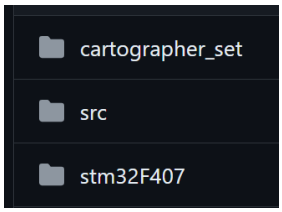
它以 stm32F407 作为运动控制中心，传感器方面采用激光雷达、IMU、编码器、摄像头。您既可以将笔记本电脑直接放置在小推车上作为 ROS 主控，也可以通过多机通信的方式控制小推车上的树莓派或其他卡片电脑，采用分布式计算的方式来减小卡片电脑的负担。

该项目成本低廉，代码风格简单易懂，老少咸宜，适合 ROS 入门学习。

文档内容包括主要器件购买链接，使用方法，ros 安装及项目移植等内容，具体条目请从文档目录进行检索。

本项目工程代码已在 github 开源:

https://github.com/JingQ-Cheng/ROS-Auto-navigation-handcart



src 是 ros 工程文件
cartographer_set 是设置 cartographer 所需的文件
stm32F407 是下位机代码

演示效果视频:

https://www.bilibili.com/video/BV1fj411D7Xz/?share_source=copy_web&vd_source=ba09f5329ccbdf0cbc6d31f78a69ca93

2.ROS 学习

关于 ROS 的学习方法可以看我之前写的文章：
https://mp.weixin.qq.com/s/w9Ok9OTw0Br0XNsjUEmdFw

欢迎关注我们的公众号，不定期分享技术学习帖：

项目的网盘资料如下：包含有本项目所需的器件资料（含激光雷达 N10、IMU）、轮趣科技以及古月老师的 ROS1 教程，学习必备！

链接：https://pan.baidu.com/s/1nnhiyLl9zRLN28b0eLjN3A?pwd=s2mw
提取码：s2mw

3.硬件说明

单片机连接说明

（本项目经过多次迭代引脚号和串口号可能有变化，具体请以工程代码为准）
cubemx 生成的代码可能会覆盖原来手动改的引脚配置，尽量直接使用 cubemx 修改引脚配置而不要手动修改。

```
* 左轮电机控制 PWM PA8 PA9
* 右轮电机控制 PWM PA10 PA11
* 左编码器用 TIM3 PA6 PA7
* 右编码器用 TIM4 PB6 PB7
```

*UART5 用于串口打印调试（非必要）

*UART6 与 PB10 PB11 与上位机通信

不想画板子可以直接购买项目同款开发板：https://m.tb.cn/h.gSixJcI?tk=ur9w3b3CXhk

电机

使用的是两个 AB 相直流有刷电机

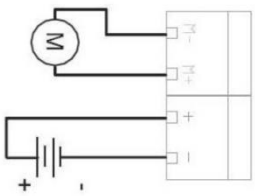
淘宝链接：
https://item.taobao.com/item.htm?spm=a1z0d.7625083.1998302264.6.5c5f4e69UZ78uS&id=45347924687
（购买的是霍尔编码器款式的）

型号	霍尔编码器	高精度GMR编码器
减速比	1:30	
减速前转速	10000rpm	
减速后转速	330rpm	
额定电流	1.44A	
额定电压	12V	
额定功率	15W	
额定扭矩	2.6kg,cm	
重量	215g	207g

电机驱动模块

需要购买两个，一个驱动器控制一个电机
淘宝链接：
https://item.taobao.com/item.htm?spm=a1z09.2.0.0.32332e8dhWceaO&id=42134661933&_u=e3hp3cij6147

TS7960B H 桥 43A 大功率电机驱动器模块概述：
该驱动器采用 BTS7960 芯片组成的大功率驱动全 H 桥驱动模块，具有热过流保护功能。
注意：该芯片是无法输出 5V 电压的，请不要用于 5V 的电机。



1、RPWM
2、LPWM
3、R_EN
4、L_EN
5、R_IS
6、L_IS
7、VCC
8、GND

连接：
方法1：
VCC：来自微控制器（MCU）的5V
GND：微控制器（MCU）的GND
L_PWM：来自微控制器（MCU）的PWM或5V用于CCW（CW）匝数
R_PWM：来自微控制器（MCU）的PWM或5V用于CW（CCW）匝数
R_EN和L_EN：来自微控制器（MCU）的5V
方法2：
VCC：来自微控制器（MCU）的5V
GND：微控制器（MCU）的GND
L_PWM：来自微控制器（MCU）的5V用于CCW（CW）匝数
R_PWM：来自微控制器（MCU）的5V用于CW（CCW）匝数

R_EN和L_EN：来自微控制器（MCU）的PWM

使用：

正常需要接入 6 根线，分别是提供基准点位的 5V 和 GND、用于使能的 R_EN 和 L_EN、以及控制正反转的 PWM 信号的 LPWM 和 RPWM。

VCC 和 R_EN 和 L_EN 都接 5V。

LPWM 和 RPWM 来自单片机 IO 口，在实际控制时，一个通入 PWM，一个置 0 便可以控制正反转。

两个 IS 脚是可以提供过流警告的，也可以不用。

上位机连接说明

上位机目前支持单板计算机和笔记本电脑以及 12V 工控机。类树莓派的单板计算机可以通过 UART 和单片机直接连接，而 PC 仅能通过 USB 转 TTL 和单片机连接。其他的激光雷达，IMU 是用 USB 进行通信。

单板计算机

使用单板计算机的方案相比于直接将笔记本电脑放在推车上，可以极大的减小推车的重量。但单板计算机的性能较为有限，对于大规模建图算力不足。一般是使用多机通信的方式，将建图算法远程运行在电脑上（区域主机），而单板计算机主要承担传感器的数据解析任务并向单片机下发控制指令。

如果要使用该种方案请网上搜索 ROS 多机通信并进行配置。

其控制架构如下图所示：

单板计算机在接口上只需要使用到单板计算机的 uart 和 USB 功能，除了树莓派也可以考虑一些国产平替，价格可能会比较便宜，如香橙派、香蕉派、nanopi 等，性能也无需太好，因为主要的算法都在电脑上运行。

本项目使用的车载单板计算机（卡片电脑）是 nanopi neo4。nanopi neo4 使用的是瑞芯微的 RK3399，内存仅有 1G，性能较低，是较为极限的使用案例。

nanopi neo4 资料：

https://wiki.friendlyelec.com/wiki/index.php/NanoPi_NEO4/zh

项目是使用 UART2 和单片机通信，在单板计算机上的映射关系为：

UART2： GPIO0 = TXD2 -> ttyAMA1 GPIO1 = RXD2 -> ttyAMA1

UART3： GPIO4 = TXD3 -> ttyAMA2 GPIO5 = RXD3 -> ttyAMA2

激光雷达、惯导和 USB 摄像头通过 USB 与卡片电脑连接。

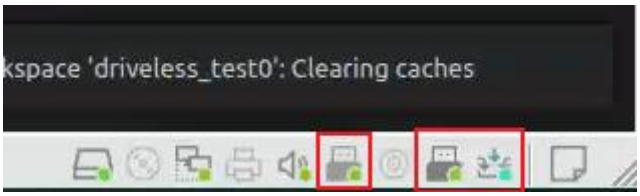
笔记本电脑/工控机

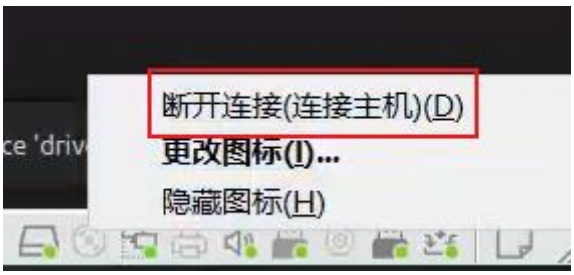
笔记本电脑有一个好处是算力比较充足，无需花额外的钱，架构相对简单，无需像卡片电脑那样进行复杂的多机通信配置。缺点是相对笨重。笔记本可以通过硬盘方式或者虚拟机方式安装 linux 系统。

如果资金充足，请使用小型工控机或小主机，可以同时兼顾性能、体积和功耗。其控制架构如下所示：

笔记本和工控机和传感器连接均通过 USB。如果您的电脑 USB 接口不足，请使用 USB 拓展坞。

虚拟机在连接设备上和电脑直连有一些区别。外设连接到电脑后，需要在虚拟机右下方开启与虚拟机连接。





注意：要先连接 cp2102USB(激光雷达)，再连接 Qinhen serial（转 TTL），连接顺序会影响串口端口号识别。激光雷达是 USB0，UART 是 USB1。

传感器

激光雷达：（400-500￥的价格）
https://item.taobao.com/item.htm?spm=a1z09.2.0.0.7df72e8deadEiX&id=661905067052&_u=e3hp3cijd427&skuId=5008765663032

IMU：（300+￥，精度较高但略为昂贵，可以选择其他产品替代）
https://item.taobao.com/item.htm?spm=a1z09.2.0.0.7df72e8deadEiX&id=641144832994&_u=e3hp3cij5927&skuId=5370718772633

100 万像素 720P USB 摄像头（20-30￥）
https://item.taobao.com/item.htm?spm=a1z09.2.0.0.7df72e8deadEiX&id=649902053428&_u=e3hp3cij82db&skuId=4687580905218

4.软件操作

车子搭建好后，按照上述硬件连接指南连接完后，所有设备通电即可使用。以下主要讲解上层 ROS 中各个功能如何使用。

在使用之前，请掌握基本的 linux 使用知识，最好对 ROS 有所了解。

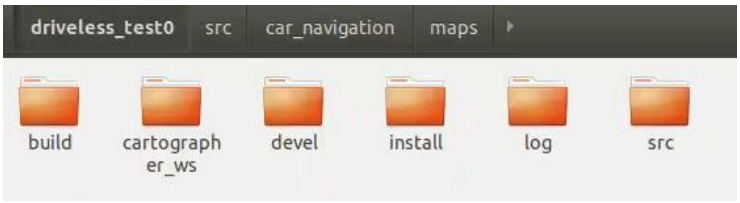
编译工程和 source
在对应工程目录下，如虚拟机中的 driveless_test0 下

```
catkin_make
```

```
source devel/setup.bash
```

上层工程目录

driveless_test0 文件夹是我们上位机代码所在。打开来可以看到：



执行大车大部分指令均在此目录下执行，你可以在该文件夹中右键打开终端，并输入相应的命令行。

在 driveless_test0 使用 roslaunch [package] [.launch] 可以执行某一项功能。比如我想执行键盘控制节点，那就在工程所在的文件夹 driveless_test0 打开命令行终端，然后输入 roslaunch keyboard keyboard_control.launch

Keyboard 是这个节点所在的功能包，而 keyboard_control.launch 是节点的 launch 文件名称。

以下介绍的是各个功能节点对应的文件夹。

src 目录

Src 目录是我们工程代码所在地，除了 src，工程目录下其他文件夹的代码都是可以通过 ROS 的 CMake 进行生成的，不需要开发人员去编写。

src 下面的每一个文件夹都对应了相应的软件包，以此来实现项目各个功能。

uart_set

通信层代码（重要），这部分代码负责上下位机通信，其中 uart_set.launch 通信层代码。如果通信发生异常，请独立运行并检查该节点的信息是否有接收或者发送的错误。

```
roslaunch uart_set uart_set.launch
```

keyboard

编写了键盘遥控小车的代码，执行 keyboard_control.launch 即可人工操控行

驶。

遥控功能使用

roslaunch uart_set uart_set.launch
roslaunch keyboard keyboard_control.launch
如果您中途重启了通信层，keyboard 同样需要重新启动才能继续使用遥控功能。

在打开 keyboard_control.launch 功能后，使用”i”进行前进,”，”进行后退，左和右分别是 j 和 l，通过 w 可以增加线速度，x 可以减少线速度。

car_navigation

自主导航和建图，编写了自主导航和建图的 launch 脚本，算法本体由 ros 自带，不在本功能包中。
在使用本文件夹下任何建图或导航功能前，都需要先启动硬件。
roslaunch car_navigation hardware.launch 启动硬件层

建图功能使用

Gamping 算法（ros 经典算法，老少咸宜）
roslaunch car_navigation gmaping_mapping.launch 使用 gmaping 算法建图
rosrun map_server map_saver -f gmapping1 保存名为 gmapping1 的地图（在工程所在的文件中可以找到）

cartographer 算法（谷歌商业化算法，成熟稳定，适合建大图，笔记本建议用这个）

cartographer 是没有包含在本工程中的，需要独立安装，建议使用小鱼一键安装中的 catographer 一键安装功能，安装后将 github 中的 cartographer_set 中的两个文件夹放入对应的位置。

cartographer_ws/src/cartographer_ros/cartographer_ros/launch/carto_star.launch

cartographer_ws/src/cartographer_ros/cartographer_ros/configuration_files/My_delta_lidar.lua

（cartographer_ws 代指我命名的 catographer 工程文件夹名称）

在 cartographer_ws 打开一个终端中执行编译：

source install_isolated/setup.bash

启动 carto 建图功能：

roslaunch cartographer_ros carto_star.launch

然后在 driveless_test0 下执行：

roslaunch car_navigation carto_slam.launch

如果您想要调整参数(传感器融合)，需要在以下路径的文件中修改
cartographer_ws/src/cartographer_ros/cartographer_ros/launch/carto_star.launch
cartographer_ws/src/cartographer_ros/cartographer_ros/configuration_files/My_delta_lidar.lua
（cartographer_ws 代指我命名的 catographer 工程文件夹名称）

目前该算法的配置是同时使用 imu 和激光，实测融合编码器后反而效果变差，具体可能因车而异。

如果你修改了该两个文件，都需要在 cartographer_ws **下执行**

catkin_make_isolated --install --use-ninja **进行编译**。

导航功能使用

在车上运行工控机/PC

roslaunch car_navigation slam_and_nav.launch 同时建图和导航（无地图下使用）

roslaunch car_navigation nav_pc.launch pc 或者工控机使用自主导航（有地图，需要在 nav_star.launch 中设置 yaml 来配置目标地图，默认是一张空地图）

多机通信，在车上运行单板机，我的单板机是 nanopi neo4
同样需要在单板机上先运行 roslaunch car_navigation hardware.launch
单板机下独立运行导航：roslaunch car_navigation nav_neo.launch 适合在，用 pc 终端 rviz 即可查看

边缘计算模式：在 PC 上运行 roslaunch car_navigation nav_edge.launch 进行算法运算。

Rviz 用法

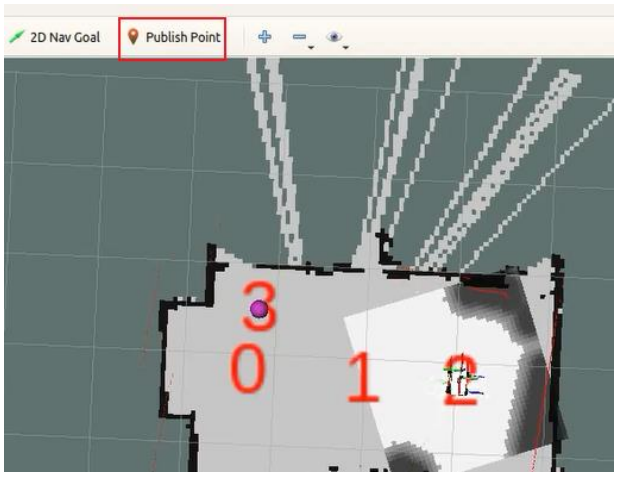
estimate：**可以设置机器人在地图中的起点位置**

拉动 2D NAV Goal 箭头是指示导航目标。

scripts

send_mark.py 用于临时多点导航功能，在 rviz 上面的按钮中使用 Publish Point。在地图上点击几个点后车子会来回在标记的几个点移动。（该节点的启动脚本命令已经集成在 nav_star.launch 中了）

其原理是 send_mark 会订阅 rviz 发布的目标点，并向 /move_base_simple/goal **最终被 move base 导航框架接收并进行相应的导航规划**。



按键 c 是清空目标点

exploring_slam.py 自主探索 slam（该功能未开启）

map

存放导航使用的地图。

send_goals

用来实现固定多点导航。利用 action 的通信格式，可以在代码中设定固定的多个目标点，一旦启动该节点，小车便会依次前往几个目标点。

使用以下命令来开启，首先要开启自主导航
roslaunch car_navigation nav_star.launch

然后用 rosrun 开启多点导航
rosrun send_goals send_goals_node

car_description

里面有 urdf 来描述车子的模型。

imu_package

imu 的 SDK，厂商提供，代码实现可以不用管。其中 ahrs_data.launch 可以用来启动 imu。

注意！！如果你在新设备第一次使用轮趣科技的 N 系列惯导，需要运行一个脚本来更改 imu 的串口识别码。(如果你获取了本项目祖传的虚拟机就不需要)，具体介绍可以看网盘资料中的惯导资料用户手册。

请在 src/imu_package 目录下打开终端，运行以下命令
sudo chmod 777 wheeltec udev.sh

sudo ./wheeltec udev.sh

lsn10

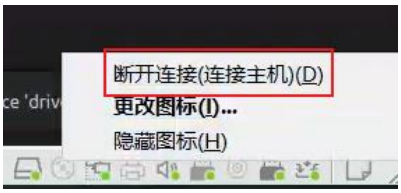
镭神 N10 单线激光雷达 SDK，由厂商提供。使用 lsn10.launch 可以开启雷达

robot_pose_ekf

扩张卡尔曼滤波融合，由 ros 官方提供。

robot_vision

编写了一些视觉代码。需要把摄像头接上，如果是虚拟机，记得在右下方让设备和虚拟机连接！



roslaunch robot_vision usb_cam.launch 开启摄像头

rqt_image_view 把画面显示出来

里面还有人脸识别，物体追踪的 CV 算法，感兴趣可以自己摸索一下。

5.将工程移植到新设备

考虑到各位可能想要在工控机或者一些卡片电脑上运行该工程，以下提供了一个简要的移植教程。

选择合适的镜像

ROS 的版本和 ubuntu 版本一一对应，每隔两年更新一个版本。ROS1 只更新到了 ubuntu20，从 ubuntu22 开始仅有 ROS2，使用新版的 ubuntu 系统无法运行 ROS1

本项目运行在 ubuntu18，所以您需要下载有 ubuntu18 及其相关衍生系统，，包括并不限于：

ubuntu18 desktop（官方桌面版，但该版本较重，不适合运行在一些卡片电脑上）

Linux mint （适用于 x86 电脑，比原生 ubuntu 更加稳定且美观）

ubuntu mate 18（一个轻量的 ubuntu 桌面，支持 X86 或树莓派）

ubuntu server（服务器版，无桌面，对性能有限的设备较为友好）

Zorin OS (类 window 桌面，x86 架构)

如果是购买了 ARM 卡片电脑，请使用开发板对应的官方 ubuntu 的发行版。

请不要下载非 ubuntu 派系的的 linux 发行版。

安装 ros

方法一

采用小鱼一键安装 ros，根据指示换源装 ros，建议直接安装完整版。ubuntu18 对应的 ros1 版本是 melodic

在终端输入 `wget http://fishros.com/install -O fishros && . Fishros`

按照提示进行安装即可，非常方便，不要再自己手动安装了 ros 了，很麻烦！

方法二（仅限于拥有本项目虚拟机）

如果你是本项目所在实验室的同学，可以找师兄复制该工程开发的用到的虚拟机。 虚拟机的主目录/code/some_package 中写了一个自动配置的脚本，如果要移植，请把 some_package 放到新设备的用户目录下，然后在新设备下的 some_package 打开终端，执行./environment.sh 就可以按照指示安装。

Shell 脚本类似于命令行，我在里面也引用了小鱼的一键安装，并且增加了其他一些内容。

可选：some_package 里还有远程桌面 todesk 的安装包，执行

sudo dpkg -i todesk-v4.3.1.0-arm64.deb 即可安装

移植工程

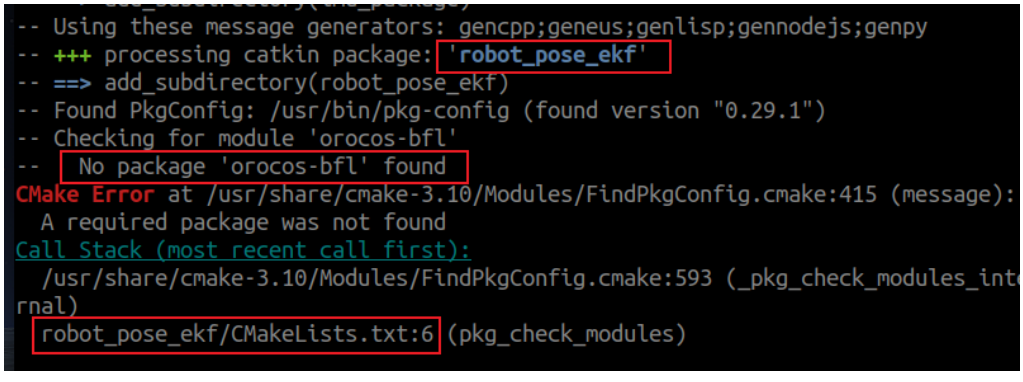
将大车目录下的 src 拷贝到新设备，找一个地方新建一个工程目录把 src 放进去，然后按照新建工程步骤编译安装。

cd /car_ws/src #进入 src 文件夹，car_ws 是工程名字，喜欢起什么名字自己起，不要带中文就好。

```
catkin_init_workspace    #初始化，改变空间的属性
cd ..                    #回到工程目录才可以进行编译
catkin_make              #编译所有功能包源码
catkin_make install              #创建安装空间
```

解决编译报错和依赖缺失

编译工程，一般 camke 会报错，一般是缺少一些依赖，只需按照 cmake 报错提示下载相应的依赖即可。缺啥就装啥，比如说 cmake 会显示 no package orocos-bfl found，我们可以看到这条错误信息是从 robot_pose_ekf 中传出来的，cmake 说他没有找到这个 package。



这说明 orocos-bfl 是 ekf 包（扩张卡尔曼滤波）的依赖，需要自行安装

sudo apt-get install liborocos-bfl-dev

其他的以此类推：

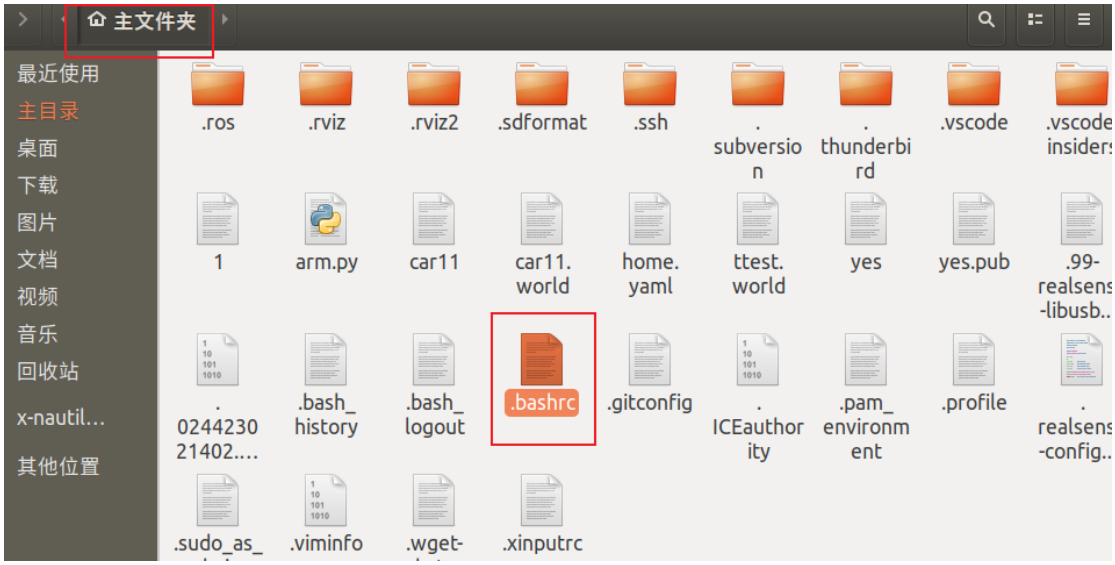
movebase 包

```
sudo apt-get install ros-melodic-navigation
```

amcl 包

```
sudo apt-get install ros-melodic-amcl
```

配置环境变量路径



我们来到主文件目录下，按 ctrl+H，将隐藏文件显示，然后点击进入.bashrc 文件，

在文件最下面添加 `source /home/passoni/driveless_test0/devel/setup.bash`

Passoni 是你的用户名，driveless_test0 是你的工程名字

然后在终端重启 bashrc

```
source ~/.bashrc
```

查看 ROS 路径是否添加上

echo $ROS_PACKAGE_PATH

这时候就不需要每次打开新终端都 source devel/setup.bash

6.Ubuntu 实用技巧

清理垃圾
Ubuntu 虚拟机的存储优化并不太好，用户在使用一段时间后会发现存储占用越来越大，甚至把电脑硬盘撑爆。以下给出了我个人的一些优化方法：

Ubuntu 常用清理命令： https://zhuanlan.zhihu.com/p/72059790
如果你安装了 vmtool 用于虚拟机和电脑传输文件，这其中的缓存也是会占用虚拟机空间，请使用以下命令行进行清理：
sudo rm -rf ~/.cache/vmware/drag_and_drop

另外，ros 打印的日志信息日积月累也会有不小的占用

rosclean check
检查日志占用的内存。

rosclean purge
清除日志信息。

虚拟机和电脑互传文件
Vmware 有自带的 vmtool，可以用来传递文件，具体安装方法请上网查。

7.STM32 端移植教程

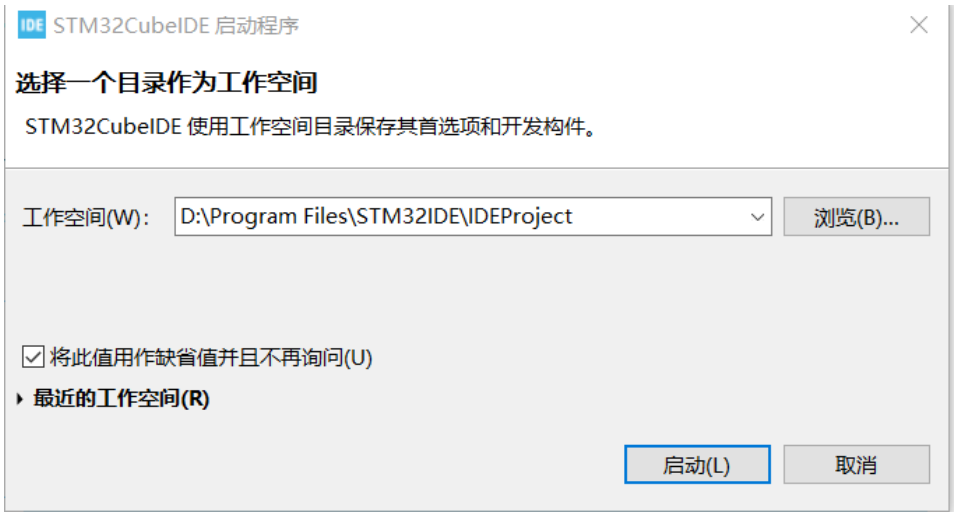
作者：赖伟俊

芯片：F407ZGT6
开发软件：STM32CubeIDE 1.10.1（ide 的尿性，随意更换版本编译会出现不明 bug）
软件汉化教程：https://blog.csdn.net/wct3344142/article/details/104142863

注意：引脚定义和串口号可能有变化，具体以工程代码为准。

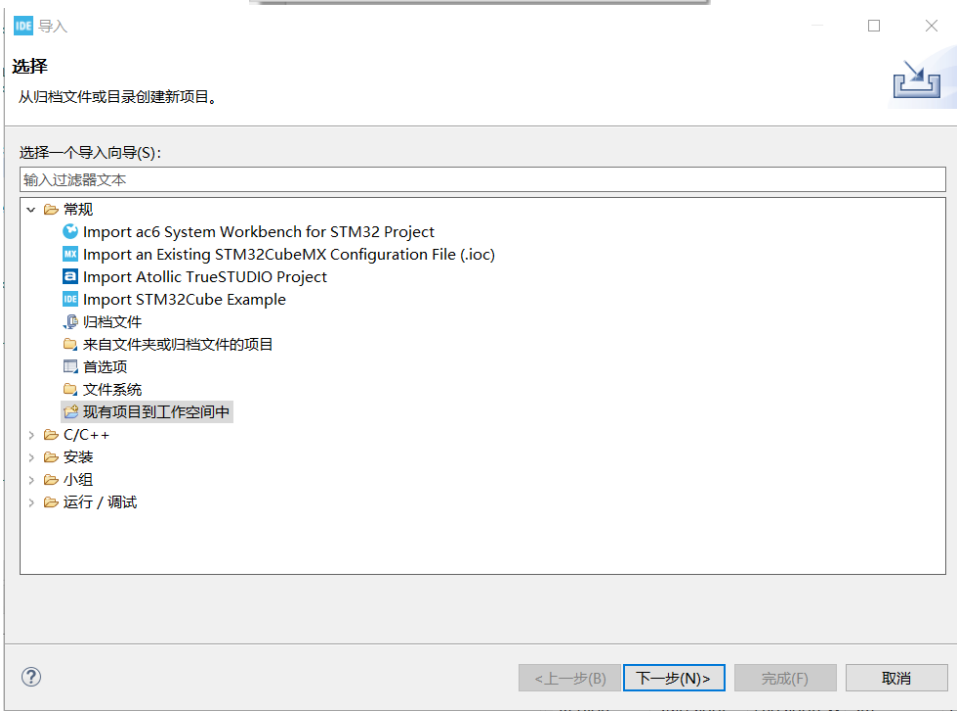
导入工程步骤

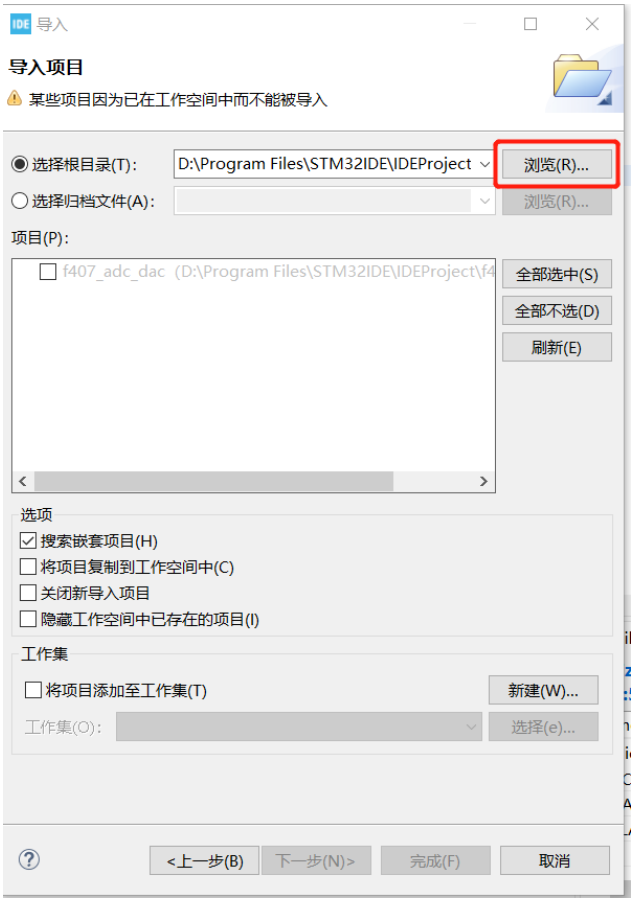
选择一个路径作为工程默认存放路径，并将工程解压后放在该路径下。不要随意更改存放路径，更改存放路径会导致某些编译器设置失效。



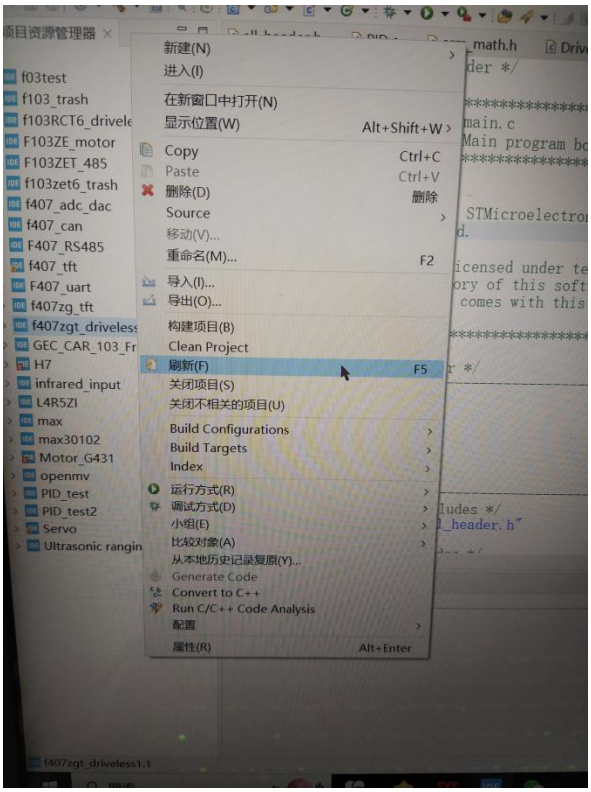
依次选择“文件” -> “导入” -> “常规” -> “现有项目到工作空间中”,选择对应的项目文件夹


18_stm32_program - STM32CubeIDE
文件(F)
编辑(E)
Source
Refactor
导航(N)
搜
新建(N) Alt+Shift+N
打开文件...
从文件系统中打开项目...
最近使用的文件
关闭(C) Ctrl+W
全部关闭(L) Ctrl+Shift+W
保存(S) Ctrl+S
另存为(A)...
全部保存(E) Ctrl+Shift+S
还原(T)
移动(V)...
重命名(M)... F2
刷新(F) F5
将行定界符转换为(V)
打印(P)... Ctrl+P
导入(I)...
导入
导出(O)...
属性(R) Alt+Enter
切换工作空间(W)
重新启动
退出(X)
CSDN @snmplink

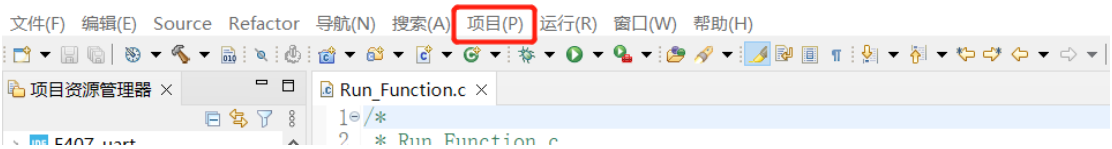

导入
选择
从归档文件或目录创建新项目。
选择一个导入向导(S):
输入过滤器文本
常规
Import ac6 System Workbench for STM32 Project
Import an Existing STM32CubeMX Configuration File (.ioc)
Import Atollic TrueSTUDIO Project
Import STM32Cube Example
归档文件
来自文件夹或归档文件的项目
首选项
文件系统
现有项目到工作空间中
C/C++
安装
小组
运行 / 调试
<上一步(B)
下一步(N)>
完成(F)
取消

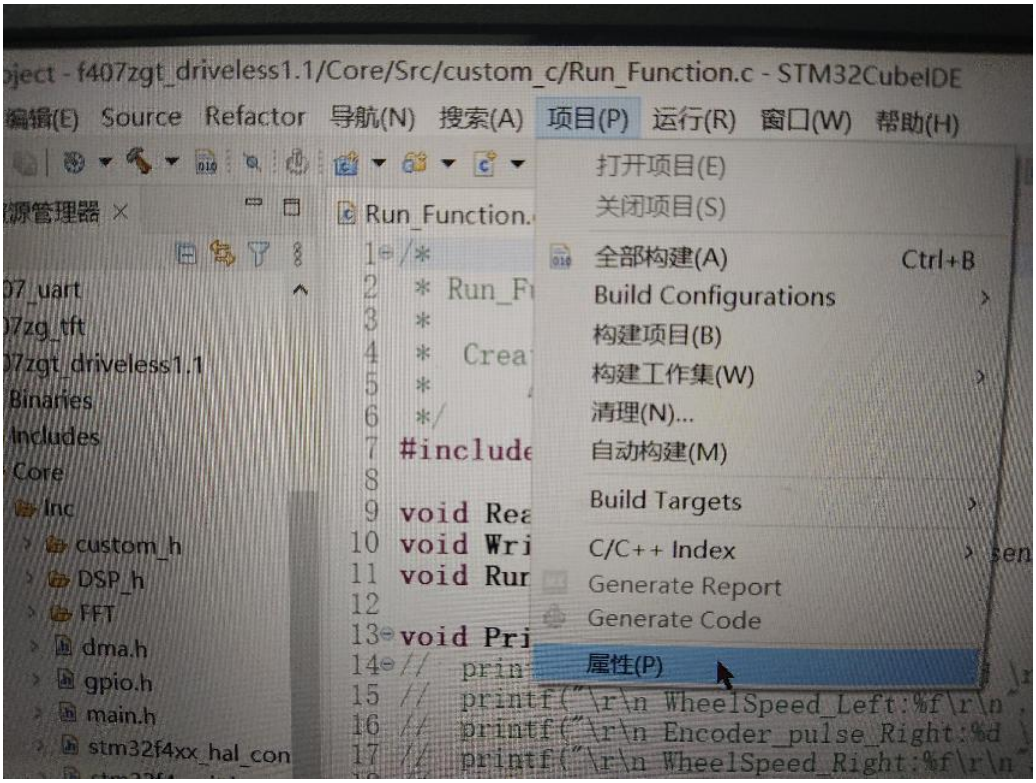


导入完成后，左侧列表如果没有出现工程，则在左侧空白处点击右键选择“刷新”

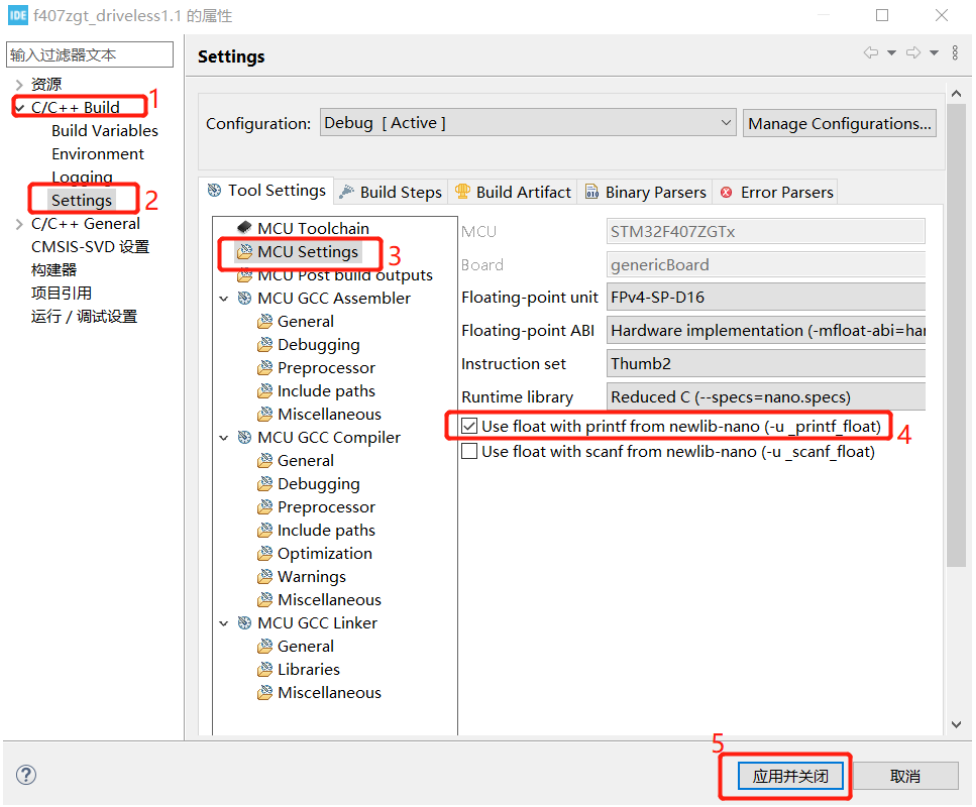


完成上述步骤后，打开 main.c，选择“项目”－>“属性”





按照如图步骤设置



移植步骤

重要提示：代码一定要写在 BEGIN 和 END 之间，否则重新生成代码后，代码不会被保存。

在 main.c 里面的开头添加**#include** "custom_h/all_header.h"这一句代码，while（1）前面增加 Driver_Init.Peripheral_Init();在 while（1）循环里面增加 Run_Function();

#include "custom_h/all_header.h"　引用所有的头文件

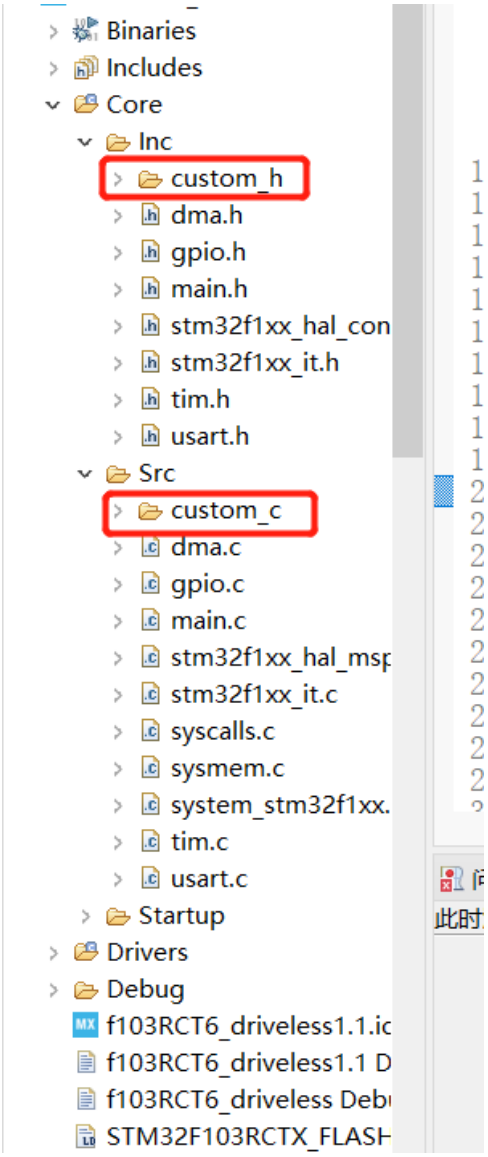
Driver_Init.Peripheral_Init();　所有功能初始化函数

Run_Function();　　主循环函数

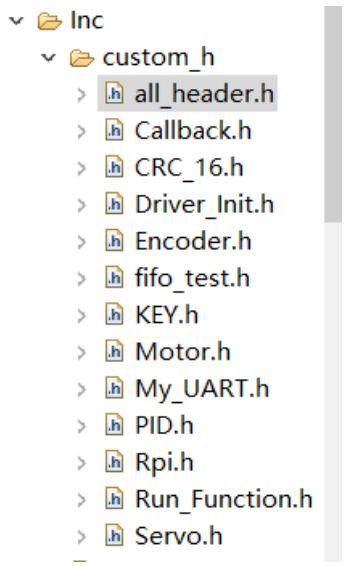
```
18 /* USER CODE END Header */
19 /* Includes ------------------------------
20 #include "main.h"
21 #include "dma.h"
22 #include "tim.h"
23 #include "usart.h"
24 #include "gpio.h"
25
26 /* Private includes ----------------------
27 /* USER CODE BEGIN Includes */
28 #include "custom_h/all_header.h"
29
30 /* USER CODE END Includes */
31
32 /* Private typedef -----------------------
33 /* USER CODE BEGIN PTD */
34
```

```
98   /* USER CODE BEGIN 2 */
99   Driver_Init.Peripheral_Init();
100  /* USER CODE END 2 */
101
102  /* Infinite loop */
103  /* USER CODE BEGIN WHILE */
104  while (1)
105  {
106    Run_Function();
107    /* USER CODE END WHILE */
108
109    /* USER CODE BEGIN 3 */
110  }
111  /* USER CODE END 3 */
112 }
```

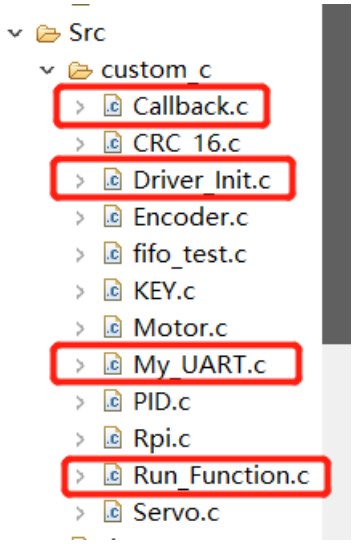
工程目录介绍



custom_h 和 custom_c 两个文件夹存放的是自定义代码，其他则是 IDE 自动生成的代码。在 IDE 配置正确的情况下只需要在自定义代码里面实现功能就可以了。



all_header.h 中存放其他所有的.h 文件，其他的文件只需要**#include** "custom_h/all_header.h"即可，如果需要新增加的.h 就把它加在这里。(所有的.h 文件都使用了#ifndef #define #endif，可以避免头文件重复引用和重定义的问题，这是 C 语言的知识，不明白的可以去搜一下，）



Driver_Init.c 放入所有的初始化
Run_Function.c 编写函数主流程
Callback.c 存放所有的中断回调函数，用来编写中断处理内容。（在中断服务函数调用的函数称为回调函数）
Ps:关于回调函数的概念，不要纠结“回调”二字，它只是个在中断里被调用的普通函数，通过__week 实现内容重定义。C 语言知识，有兴趣就去了解一下
My_UART.c 实现 printf 打印功能，可以指向不同串口（该串口要先在 CUBEIDE 里面打开）。

```
HAL_UART_Transmit(&huart5, (uint8_t *)&ch, 1, 0xFFFF);
return ch;
```